EXHIBIT 1

EXHIBIIT 1

PSEUDO CODE LISTING OF FOLLOW UP FOR ASTHMA OFFICE VISIT DIALOG

```
1.  package com.solvetech.st_notify;

2.  import java.util.*;

3.  import com.solvetech.st_lang.*;

4.  public class FU_Asthma_Dialog extends DialogBase
5.  {
6.      private static final String ATTR_DOCTOR_USES_PFR1          = "person(doctor).uses_pfr1";
7.      private static final String ATTR_DOCTOR_USES_PFR2          = "person(doctor).uses_pfr2";

8.      // is the inOfficePF an attribute of the patient or of the dialog??
9.      private static final String ATTR_PATIENT_INOFFICEPF        = "person(patient).inOfficePF";

10.     private static final String ATTR_PATIENT_AGE = "person(patient).age";
11.     private static final String ATTR_PATIENT_HEIGHT    = "person(patient).height";
12.     private static final String ATTR_PATIENT_SEX  = "person(patient).sex";

13.     private final static String ATTR_DIALOG_WORSE              = "dialog.worse";
14.     private final static String ATTR_DIALOG_READY              = "dialog.ready";
15.     private final static String ATTR_DIALOG_RETRIG             = "dialog.retrig";
16.     private final static String ATTR_DIALOG_RECALL_1       = "dialog.recall_1";
17.     private final static String ATTR_DIALOG_RECALL_2       = "dialog.recall_2";
18.     private final static String ATTR_DIALOG_OK_END             = "dialog.okend";
```

```
19.     private final static String ATTR_DIA LOG_ACK_ER          = "dialog.ackER";
20.     private final static String ATTR_DIALOG_PF_1          = "dialog.pf_1";

21.     //=================================================
22.     private boolean doctor_uses_pfr1;
23.     private boolean doctor_uses_pfr2;
24.     private int inOfficePF;
25.     private int retrig;
26.     private int age = 0;
27.     private int height = 0;
28.     private String sex = "";
29.
30.     public FU_Asthma_Dialog()
31.     {
32.         super();
33.     }

34.     final protected void init()
35.         throws Exception
36.     {
37.         // load up all the variables passed in from the message.
38.         doctor_uses_pfr1  = getAttrBoolean(ATTR_DOCTOR_USES_PFR1);
39.         doctor_uses_pfr2  = getAttrBoolean(ATTR_DOCTOR_USES_PFR2);
40.         inOfficePF              = getAttrInt(ATTR_PATIENT_INOFFICEPF);
41.         retrig                  = getAttrInt(ATTR_DIALOG_RETRIG);
42.         age                         = getAttrInt(ATTR_PATIENT_AGE);
```

```
43.         height                  = getAttrInt(ATTR_PATIENT_HEIGHT);
44.         sex                     = getAttrString(ATTR_PATIENT_SEX);
45.     }

46.     final protected void register()
47.     {
48.         // registration is a way of telling scheduler what attributes are
49.         // needed for a message.
50.         registerAttribute(ATTR_DOCTOR_USES_PFR1);
51.         registerAttribute(ATTR_DOCTOR_USES_PFR2);
52.         registerAttribute(ATTR_PATIENT_INOFFICEPF);
53.         registerAttribute(ATTR_PATIENT_AGE);
54.         registerAttribute(ATTR_PATIENT_HEIGHT);
55.         registerAttribute(ATTR_PATIENT_SEX);
56.         registerAttribute(ATTR_DIALOG_WORSE);
57.         registerAttribute(ATTR_DIALOG_READY);
58.         registerAttribute(ATTR_DIALOG_RETRIG);
59.         registerAttribute(ATTR_DIALOG_RECALL_1);
60.         registerAttribute(ATTR_DIALOG_RECALL_2);
61.         registerAttribute(ATTR_DIALOG_OK_END);
62.         registerAttribute(ATTR_DIALOG_ACK_ER);
63.         registerAttribute(ATTR_DIALOG_PF_1);

64.         // register the Prompts that we are using.  So that
65.         // they can be verified before the conversation begins.
66.         registerPrompt(PROMPT_ID_GOTO_ER);
```

```
67.             registerPrompt(PROMPT_ID_OK_END);
68.             registerPrompt(PROMPT_ID_PEAK_FLOW);
69.             registerPrompt(PROMPT_ID_READY_PF1);
70.             registerPrompt(PROMPT_ID_RECALL_30);
71.             registerPrompt(PROMPT_ID_WORSE);

72.             registerPrompt(PROMPT_ID_ALERT_WORSE);
73.             registerPrompt(PROMPT_ID_ALERT_NO_RESPONSE);
74.             registerPrompt(PROMPT_ID_ALERT_PF1_BELOW_OFFICE_PF);
75.             registerPrompt(PROMPT_ID_ALERT_PF1_BELOW_50_PERCENT_PREDICTED);
76.             registerPrompt(PROMPT_ID_ALERT_STAT_CHECK);
77.
78.             // register the roles that we require
79.             registerRole(ROLE_PATIENT);
80.             registerRole(ROLE_DOCTOR);
81.         }

82.         final protected void term()
83.                 throws Exception
84.         {
85.         }

86.     final protected void execute()
87.     {
88.             Phrase hello = new Phrase();
89.             hello.addMessage("Hello! Are you, ");
```

```
90.     String patientName = getPatientName();
91.     hello.addName(patientName);
92.     hello.addMessage("? ");
93.     int areYouTheOne = askYesNo(hello);
94.     if(areYouTheOne != ASK_RESPONSE_YES)
95.     {
96.         setStatus(DIALOG_STATUS_NORESPONSE);
97.         return;
98.     }

99.     Phrase messageFrom = new Phrase();
100.    messageFrom.addMessage("There is a message from your doctor, ");
101.    String doctorName = getDoctorName();
102.    messageFrom.addName(doctorName);
103.    messageFrom.addMessage(", recorded at");
104.    messageFrom.addTime();
105.    if(!getAcknowledgement(messageFrom))
106.    {
107.        setStatus(DIALOG_STATUS_FAIL);
108.        return;
109.    }
110.
111.    if(retrig <= 0)
112.    {
113.            // [D1Q1] first question
114.            int worse = askYesNo(PROMPT_ID_WORSE);
```

```
115.                    saveAnswer(ATTR_DIALOG_WORSE,worse);

116.                    if(worse == ASK_RESPONSE_YES)
117.                            alertDoctor(PROMPT_ID_ALERT_WORSE);
118.                    else if(worse == ASK_RESPONSE_NONE)
119.                    {
120.                            alertDoctor(PROMPT_ID_ALERT_NO_RESPONSE);
121.                            return;
122.                    }
123.            }

124.            // [D1Q2] second question
125.            int ready = askYesNo(PROMPT_ID_READY_PF1);
126.            saveAnswer(ATTR_DIALOG_READY,ready);
127.            if(ready != ASK_RESPONSE_YES)
128.            {
129.                    retrig++;
130.                    switch(retrig)
131.                    {
132.                      case 1:
133.                       {
134.                         saveAnswer(ATTR_DIALOG_RETRIG,retrig);
135.
saveAnswer(ATTR_DIALOG_RECALL_1,getAcknowledgement(PROMPT_ID_RECALL_30));
136.                         callAgainMinutes(30);
137.                         break;
```

```
138.                    }
139.                  case 2:
140.                   {
141.                    saveAnswer(ATTR_DIALOG_RETRIG,retrig);
142.
    saveAnswer(ATTR_DIALOG_RECALL_2,getAcknowledgement(PROMPT_ID_RECALL_30));
143.                    callAgain(30,TIME_UNITS_MINUTES);
144.                    break;
145.                    }
146.                   case 3:
147.                    {
148.                     alertDoctor(PROMPT_ID_ALERT_NO_PF1);
149.                     setStatus(DIALOG_STATUS_UNKNOWN);
150.                     break;
151.                     }
152.                }
153.               return;
154.          }

155.          // [D1Q5] third question
156.          int pf = askInt(PROMPT_ID_PEAK_FLOW);
157.          saveAnswer(ATTR_DIALOG_PF_1,pf);
158.          if(peakflow_rule1(pf) && doctor_uses_pfr1)
159.          {
160.                alertDoctor(PROMPT_ID_ALERT_PF1_BELOW_OFFICE_PF);
161.          }
```

```
162.            else if(peakflow_rule2(pf) && doctor_uses_pfr2)
163.            {
164.                    alertDoctor(PROMPT_ID_ALERT_PF1_BELOW_50_PERCENT_PREDICTED);
165.            }
166.            else
167.            {
168.                    // [D1Q7] - ready to do dialog 2 on next scheduling time.
169.                    saveAnswer(ATTR_DIALOG_OK_END,
    getAcknowledgement(PROMPT_ID_OK_END));
170.                    return;
171.            }

172.            // [D1Q6]
173.            saveAnswer(ATTR_DIALOG_ACK_ER, getAcknowledgement(PROMPT_ID_GOTO_ER));
174.            alertDoctor(PROMPT_ID_ALERT_STAT_CHECK);
175.    }

176.    private boolean peakflow_rule1(int pf)
177.    {
178.            if(pf < inOfficePF)
179.                    return true;

180.            return false;
181.    }
182.
183.    // compare peak flow to a formula to compute pf based on height, age, sex
```

```
184.        private boolean peakflow_rule2(int pf)
185.        {
186.                double predicted_pf = 0;

187.                if(sex.equalsIgnoreCase("male"))
188.                {
189.                        predicted_pf = ((((double)height * 0.0254 * 5.48) + 1.58) - ((double)age * 0.041)) *
    60.0;
190.                }
191.                else
192.                {
193.                        predicted_pf = ((((double)height * 0.0254 * 3.72) + 2.24) - ((double)age * 0.03)) * 60.0;
194.                }
195.

196.                double percent = ((double)pf * 100.0) / predicted_pf;
197.                if(percent < 50.0)
198.                        return true;

199.                return false;
200.        }
201. }
```

EXHIBIT 2

EXHIBIT 2: PSEUDO CODE LISTING FOR CHEMISTRY BATTERY TEST RESULT DIALOG

```
package com.solvetech.st_notify;
import java.util.*,  com.solvetech.st_lang.*;

public class ChemicalBatteryTest_Dialog extends DialogBase
{
        private static final int COMPARE_TEST_NORMAL                    = 0;
        private static final int COMPARE_TEST_LOW                       = -1;
        private static final int COMPARE_TEST_HIGH                      = 1;
        private static final int COMPARE_TEST_INDETERMINATE             = -2;

        // attributes
        private static final String ATTR_PATIENT_AGE   = "person(patient).age";
        private static final String ATTR_PATIENT_SEX   = "person(patient).sex";

        // constants for test results
        private static final int TEST_FIELD_NAME                    =   0;
        private static final int TEST_FIELD_RESULT                  =   1;
        private static final int TEST_FIELD_NORMAL_LOW              =   2;
        private static final int TEST_FIELD_NORMAL_HIGH             =   3;
        private static final int TEST_FIELD_UNITS                   =   4;
        private static final int TEST_FIELD_REFERENCE_NORMAL        =   5;
        private static final int TEST_FIELD_REFERENCE_LOW           =   6;
        private static final int TEST_FIELD_REFERENCE_HIGH          =   7;
        private static final int TEST_FIELD_REFERENCE_INDETERMINATE = 8;

        private static final int TEST_OUT_NAME              =   0;
        private static final int TEST_OUT_RESULT                =   1;
        private static final int TEST_OUT_NORMAL_LOW        =   2;
        private static final int TEST_OUT_NORMAL_HIGH       =   3;
```

```
31      private static final int TEST_OUT_YOUR_RESULT      =      4;
32      // computed - not sent
33      private static final int TEST_OUT_UNITS            =      5;
34      private static final int TEST_OUT_REFERENCE        =      6;
35      private static final int TEST_OUT_MAX              =      7;
36
37      private static final int TEST_TERSE_OUT_NAME       =      0;
38      private static final int TEST_TERSE_OUT_RESULT     =      1;
39      private static final int TEST_TERSE_OUT_UNITS      =      2;
40      private static final int TEST_TERSE_OUT_MAX        =      3;
41
42      private static final String TEST_COLUMN_YOUR_RESULT = "yourResult";
43      private static final String TEST_COLUMN_REFERENCE = "reference";
44
45
46      // headers
47      // these must stay in sync with the TEST_FIELD_... constants defined above
48      private static String[] TEST_HEADER_NAMES =
49      {
50              "dialog.test.name",
51              "dialog.test.result",
52              "dialog.test.normal_low",
53              "dialog.test.normal_high",
54              "dialog.test.units",
55              "dialog.test.reference_normal",
56              "dialog.test.reference_low",
57              "dialog.test.reference_high",
58              "dialog.test.reference_indeterminate"
59      };
60
61
```

```
62      // indexes into array TEST_ATTR_FIELD_NAMES
63
64      private static final int TEST_INDEX_GLUCOSE                 = 0;
65      private static final int TEST_INDEX_UREA_NITROGEN           = 1;
66      private static final int TEST_INDEX_CREATININE              = 2;
67      private static final int TEST_INDEX_CALCIUM                 = 3;
68      private static final int TEST_INDEX_SODIUM                  = 4;
69      private static final int TEST_INDEX_POTASSIUM               = 5;
70      private static final int TEST_INDEX_CO2                     = 6;
71      private static final int TEST_INDEX_CHLORIDE                = 7;
72      private static final int TEST_INDEX_TOTAL_PROTEIN           = 8;
73      private static final int TEST_INDEX_ALBUMIN                 = 9;
74      private static final int TEST_INDEX_GLOBULIN                = 10;
75      private static final int TEST_INDEX_A_G_RATIO               = 11;
76      private static final int TEST_INDEX_TOTAL_BILIRUBIN         = 12;
77      private static final int TEST_INDEX_ALKALINE_PHOSPHATASE    = 13;
78      private static final int TEST_INDEX_AST_SGOT                = 14;
79      private static final int TEST_INDEX_AST_SGPT                = 15;
80
81
82      // Table of attribute names - this will be used to fetch
83      // the values of the attributes so that the results can be reported.
84      private static String[][] TEST_ATTR_FIELD_NAMES =
85      {
86              {       "dialog.test.name.glucose",
87                      "dialog.test.result.glucose",
88                      "dialog.test.normal_low.glucose",
89                      "dialog.test.normal_high.glucose",
90                      "dialog.test.units.glucose",
91                      "dialog.test.reference_normal.glucose",
92                      "dialog.test.reference_low.glucose",
```

```
93              "dialog.test.reference_high.glucose",
94              "dialog.test.reference_indeterminate.glucose"
95          },
96
97          {   "dialog.test.name.urea_nitrogen",
98              "dialog.test.result.urea_nitrogen",
99              "dialog.test.normal_low.urea_nitrogen",
100             "dialog.test.normal_high.urea_nitrogen",
101             "dialog.test.units.urea_nitrogen",
102             "dialog.test.reference_normal.nitrogen",
103             "dialog.test.reference_low.nitrogen",
104             "dialog.test.reference_high.nitrogen",
105             "dialog.test.reference_indeterminate.nitrogen"
106         },
107
108         {   "dialog.test.name.creatinine",
109             "dialog.test.result.creatinine",
110             "dialog.test.normal_low.creatinine",
111             "dialog.test.normal_high.creatinine",
112             "dialog.test.units.creatinine",
113             "dialog.test.reference_normal.creatinine",
114             "dialog.test.reference_low.creatinine",
115             "dialog.test.reference_high.creatinine",
116             "dialog.test.reference_indeterminate.creatinine"
117         },
118
119         {   "dialog.test.name.calcium",
120             "dialog.test.result.calcium",
121             "dialog.test.normal_low.calcium",
122             "dialog.test.normal_high.calcium",
123             "dialog.test.units.calcium",
```

```
124                 "dialog.test.reference_normal.calcium",
125                 "dialog.test.reference_low.calcium",
126                 "dialog.test.reference_high.calcium",
127                 "dialog.test.reference_indeterminate.calcium"
128             },
129
130             {   "dialog.test.name.sodium",
131                 "dialog.test.result.sodium",
132                 "dialog.test.normal_low.sodium",
133                 "dialog.test.normal_high.sodium",
134                 "dialog.test.units.sodium",
135                 "dialog.test.reference_normal.sodium",
136                 "dialog.test.reference_low.sodium",
137                 "dialog.test.reference_high.sodium",
138                 "dialog.test.reference_indeterminate.sodium"
139             },
140
141
142             {   "dialog.test.name.potassium",
143                 "dialog.test.result.potassium",
144                 "dialog.test.normal_low.potassium",
145                 "dialog.test.normal_high.potassium",
146                 "dialog.test.units.potassium",
147                 "dialog.test.reference_normal.potassium",
148                 "dialog.test.reference_low.potassium",
149                 "dialog.test.reference_high.potassium",
150                 "dialog.test.reference_indeterminate.potassium"
151             },
152
153             {   "dialog.test.name.co2",
154                 "dialog.test.result.co2",
```

```
155                 "dialog.test.normal_low.co2",
156                 "dialog.test.normal_high.co2",
157                 "dialog.test.units.co2",
158                 "dialog.test.reference_normal.co2",
159                 "dialog.test.reference_low.co2",
160                 "dialog.test.reference_high.co2",
161                 "dialog.test.reference_indeterminate.co2"
162         },
163
164
165         {       "dialog.test.name.chloride",
166                 "dialog.test.result.chloride",
167                 "dialog.test.normal_low.chloride",
168                 "dialog.test.normal_high.chloride",
169                 "dialog.test.units.chloride",
170                 "dialog.test.reference_normal.chloride",
171                 "dialog.test.reference_low.chloride",
172                 "dialog.test.reference_high.chloride",
173                 "dialog.test.reference_indeterminate.chloride"
174         },
175
176
177         {       "dialog.test.name.total_protein",
178                 "dialog.test.result.total_protein",
179                 "dialog.test.normal_low.total_protein",
180                 "dialog.test.normal_high.total_protein",
181                 "dialog.test.units.total_protein",
182                 "dialog.test.reference_normal.total_protein",
183                 "dialog.test.reference_low.total_protein",
184                 "dialog.test.reference_high.total_protein",
185                 "dialog.test.reference_indeterminate.total_protein"
```

```
186             },
187
188
189             {       "dialog.test.name.albumin",
190                     "dialog.test.result.albumin",
191                     "dialog.test.normal_low.albumin",
192                     "dialog.test.normal_high.albumin",
193                     "dialog.test.units.albumin",
194                     "dialog.test.reference_normal.albumin",
195                     "dialog.test.reference_low.albumin",
196                     "dialog.test.reference_high.albumin",
197                     "dialog.test.reference_indeterminate.albumin"
198             },
199
200             {       "dialog.test.name.globulin",
201                     "dialog.test.result.globulin",
202                     "dialog.test.normal_low.globulin",
203                     "dialog.test.normal_high.globulin",
204                     "dialog.test.units.globulin",
205                     "dialog.test.reference_normal.globulin",
206                     "dialog.test.reference_low.globulin",
207                     "dialog.test.reference_high.globulin",
208                     "dialog.test.reference_indeterminate.globulin"
209             },
210
211
212             {       "dialog.test.name.a_g_ratio",
213                     "dialog.test.result.a_g_ratio",
214                     "dialog.test.normal_low.a_g_ratio",
215                     "dialog.test.normal_high.a_g_ratio",
216                     "dialog.test.units.a_g_ratio",
```

```
217                 "dialog.test.reference_normal.a_g_ratio",
218                 "dialog.test.reference_low.a_g_ratio",
219                 "dialog.test.reference_high.a_g_ratio",
220                 "dialog.test.reference_indeterminate.a_g_ratio"
221             },
222
223
224             {   "dialog.test.name.total_bilirubin",
225                 "dialog.test.result.total_bilirubin",
226                 "dialog.test.normal_low.total_bilirubin",
227                 "dialog.test.normal_high.total_bilirubin",
228                 "dialog.test.units.total_bilirubin",
229                 "dialog.test.reference_normal.total_bilirubin",
230                 "dialog.test.reference_low.total_bilirubin",
231                 "dialog.test.reference_high.total_bilirubin",
232                 "dialog.test.reference_indeterminate.total_bilirubin"
233             },
234
235
236             {   "dialog.test.name.alkaline_phosphatase",
237                 "dialog.test.result.alkaline_phosphatase",
238                 "dialog.test.normal_low.alkaline_phosphatase",
239                 "dialog.test.normal_high.alkaline_phosphatase",
240                 "dialog.test.units.alkaline_phosphatase",
241                 "dialog.test.reference_normal.alkaline_phosphatase",
242                 "dialog.test.reference_low.alkaline_phosphatase",
243                 "dialog.test.reference_high.alkaline_phosphatase",
244                 "dialog.test.reference_indeterminate.alkaline_phosphatase"
245             },
246
247             {   "dialog.test.name.ast_sgot",
```

```
248                     "dialog.test.result.ast_sgot",
249                     "dialog.test.normal_low.ast_sgot",
250                     "dialog.test.normal_high.ast_sgot",
251                     "dialog.test.units.ast_sgot",
252                     "dialog.test.reference_normal.ast_sgot",
253                     "dialog.test.reference_low.ast_sgot",
254                     "dialog.test.reference_high.ast_sgot",
255                     "dialog.test.reference_indeterminate.ast_sgot"
256             },
257
258
259             {       "dialog.test.name.alt_sgpt",
260                     "dialog.test.result.alt_sgpt",
261                     "dialog.test.normal_low.alt_sgpt",
262                     "dialog.test.normal_high.alt_sgpt",
263                     "dialog.test.units.alt_sgpt",
264                     "dialog.test.reference_normal.alt_sgpt",
265                     "dialog.test.reference_low.alt_sgpt",
266                     "dialog.test.reference_high.alt_sgpt",
267                     "dialog.test.reference_indeterminate.alt_sgpt"
268             }
269     };
270
271
272     private static final String TEST_EXPLANATION =
273             "The Chemical Screening Battery consists of a group of tests of "+
274             "liver, kidney, and pancreas functioning as well as other blood tests" +
275             "that are important for health such as calcium, total protein, "+
276             "and albumin." + "\n" +
277             "    The fasting blood glucose can indicate how well the pancreas is "+
278             "producing insulin; high levels are associated with diabetes...";
```

```
279         //.
280
281         // The following two tables contain explanations for abnormal results for
282         // each test.  The first table applies to high test results; the second
283         // to lower than normal results.
284         // It must be kept in the same order as the TEST_INDEX constants
285         // defined above.
286
287         private static final String[] ABNORMAL_HIGH_TEST_EXPLANATION =
288         {
289                 //      TEST_ INDEX_ HIGH_GLUCOSE
290                 "Elevated blood glucose can be the result of Diabetes Mellitus.  If this has not been noted before, you should
291 discuss this result with your medical provider",
292
293                 //      TEST_ INDEX_ HIGH_UREA_NITROGEN
294                 "High urea nitrogen can be associated with dehydration,  kidney problems, or the use of certain medicines",
295
296
297                 //      TEST_INDEX_ HIGH_CREATININE
298                 "High creatinine can be associated with dehydration,  kidney problems, or the use of certain medicines ",
299
300                 //      TEST_INDEX_ HIGH_CALCIUM
301                 "Elevated calcium can be associated with parathyroid gland disorders, certain medications and use of excess
302 vitamin D, kidney problems, and other disorders",
303                 //      TEST_INDEX_ HIGH_SODIUM
304                 "High sodium levels can result from dehydration, diarrhea, excess salt intake, and other disorders",
305
306                 //      TEST_INDEX_ HIGH_POTASSIUM
307                 "High potassium levels can come from excess potassium ingestion, renal problems, certain medications, and other
308 disorders",
309
```

```
310                 //      TEST_INDEX_ HIGH_CO2
311                 "Elevated carbon dioxide levels can come from repeated vomiting, emphysema, and certain medications",
312
313
314
315                 //      TEST_INDEX_ HIGH_CHLORIDE
316                 "Elevated chloride levels may occur with dehydration, parathyroid disease, adrenal disease, and certain metabolic
317   disorders",
318
319                 //      TEST_INDEX_ HIGH_TOTAL_PROTEIN
320                 "High protein levels can result from dehydration, liver disease, chronic infectons, and chronic inflammatory
321   conditions",
322
323                 //      TEST_INDEX_ HIGH_ALBUMIN
324                 "Albumin values can be elevated in dehydration states",
325
326                 //      TEST_INDEX_ HIGH_GLOBULIN
327                 "High globulin levels can occur in chronic infections, liver disease, or auto-immune diseases",
328
329                 //      TEST_INDEX_A_G_ HIGH_RATIO
330                 "Please see comments regarding albumin and globulin",
331
332                 //      TEST_INDEX_ HIGH_TOTAL_BILIRUBIN
333                 "Elevated bilirubin levels can result from liver disease or breakdown of red blood cells",
334
335
336                 //      TEST_INDEX_ HIGH_ALKALINE_PHOSPHATASE
337                 "Elevated levels of alkaline phosphatase occur with liver problems, gall bladder disease, and infectious
338    mononucleosis",
339
340                 //      TEST_INDEX_ HIGH_AST_SGOT
```

```
341                 "Elevated AST levels occur in liver disease and infectious
342  mononucleosis",
343                 //      TEST_INDEX_ HIGH_ALT_SGPT
344                 "Increased ALT levels can occur in liver or pancreatic disease, heart or muscle disease, and infectious
345  mononucleosis "
346         };
347
348
349         // These entries must be kept in the same order as the
350         // TEST_INDEX constants defined above.
351
352         private static final String[] ABNORMAL_LOW_TEST_EXPLANATION =
353         {
354                 //      TEST_ INDEX_ LOW_GLUCOSE
355                 "Low blood glucose can be caused by rare pancreatic or metabolic disorders or certain medications",
356
357
358                 //      TEST_ INDEX_ LOW_UREA_NITROGEN
359                 "Low urea nitrogen can be associated with poor protein intake or excess fluid intake ",
360
361
362                 //      TEST_INDEX_ LOW_CREATININE
363                 " Low creatinine can be associated with decrease muscle mass",
364
365                 //      TEST_INDEX_ LOW_CALCIUM
366                 "Low calcium levels can be associated with parathyroid gland disorders, vitamin D deficiency, kidney problems,
367  and other disorders",
368
369                 //      TEST_INDEX_ LOW_SODIUM
370                 "Low sodium levels can result from over hydration, diarrhea, diuretic and other medications, and other disorders",
371
```

```
372                 //      TEST_INDEX_ LOW_POTASSIUM
373                 "Low potassium levels can result from adrenal disorders, certain medications such as diuretics, diabetes, and other
374   disorders",
375
376                 //      TEST_INDEX_ LOW_CO2
377                 "Low carbon dioxide levels can result from diarrhea, kidney problems, diabetes, and certain medications",
378
379                 //      TEST_INDEX_ LOW_CHLORIDE
380                 "Low chloride levels can result from vomiting, diabetes, decreased adrenal function, and certain medications",
381
382
383                 //      TEST_INDEX_ LOW_TOTAL_PROTEIN
384                 "Low protein levels can result from liver disease, malabsorption, and poor nutrition",
385
386                 //      TEST_INDEX_ LOW_ALBUMIN
387                 "Low albumin levels can result from liver disease, malabsorption conditions, and poor nutrition",
388
389                 //      TEST_INDEX_ LOW_GLOBULIN
390                 "Low globulin levels can occur with renal disease and certain congenital or acquired immune disorders, ",
391
392                 //      TEST_INDEX_A_G_ LOW_RATIO
393                 "Low A/G ratios can result from liver disease, collagen disease, chronic infections, and poor nutrition",
394
395                 //      TEST_INDEX_ LOW_TOTAL_BILIRUBIN
396                 "This is a liver test; low levels of bilirubin are not worrisome",
397
398
399                 //      TEST_INDEX_ LOW_ALKALINE_PHOSPHATASE
400                 "Low levels of alkaline phosphatase can occur with poor nutrition, hypothyroidism, severe anemia, and
401   hypophosphatasia",
402
```

```
403             //      TEST_INDEX_ LOW_AST_SGOT
404             "This is a liver function test; low levels of AST are not worrisome",
405
406             //      TEST_INDEX_ LOW_ALT_SGPT
407             " This is a liver function test; low levels of ALT are not worrisome"
408         };
409
410         //================================================
411         // the header names will contain the names of each column
412         // in the table of test results.
413         private String[] headerNames;
414
415         // the testRows contain the data values corresponding to the
416         // array of attribute constants, TEST_ATTR_FIELD_NAMES, defined above
417         // each row represents a different test.
418         // for instance testRows[3] would represent the values for the calium
419         // test.  The index, TEST_INDEX_CALCIUM, would be used to address
420         // this row, ie. testRows[TEST_INDEX_CALCIUM].  The values of a row
421         // would correspond (in order) to the test name, result, normal_low, normal_high,
422         // units and a reference.  For instance testRows[TEST_INDEX_GLUCOSE][TEST_FIELD_RESULT]
423         // would contain the patients test result value for the glucose test.
424
425         private String[][] testRows;
426
427         // the results and normals are separated out so that they can be
428         // used in calculations.  They are loaded up from the attribute information.
429         private double[] testResult;
430         private double[] testNormalLow;
431         private double[] testNormalHigh;
432
433         private String sex = "";
```

```
434     private int age = 0;
435
436     public ChemicalBatteryTest_Dialog()
437     {
438         super();
439     }
440
441     final protected void init()
442         throws Exception
443     {
444         headerNames = new String[TEST_HEADER_NAMES.length];
445         testRows    = new String[TEST_ATTR_FIELD_NAMES.length][];
446
447         // load up all the variables passed in from the message.
448         // load the headers
449
450         for(int i=0;i<TEST_HEADER_NAMES.length;i++)
451         {
452             headerNames[i] = getAttrString(TEST_HEADER_NAMES[i]);
453         }
454
455         for(int i=0;i<TEST_ATTR_FIELD_NAMES.length;i++)
456         {
457             String[] row = TEST_ATTR_FIELD_NAMES[i];
458             testRows[i] = new String[row.length];
459
460             for(int j=0;j<testRows[i].length;j++)
461             {
462                 testRows[i][j] = getAttrString(TEST_ATTR_FIELD_NAMES[i][j]);
463             }
464             testResult[i]= getAttrDouble(TEST_ATTR_FIELD_NAMES[i][TEST_FIELD_RESULT]);
```

```
465                         testNormalLow[i]     =
466 getAttrDouble(TEST_ATTR_FIELD_NAMES[i][TEST_FIELD_NORMAL_LOW]);
467                         testNormalHigh[i]    =
468 getAttrDouble(TEST_ATTR_FIELD_NAMES[i][TEST_FIELD_NORMAL_HIGH]);
469                 }
470
471                 age = getAttrInt(ATTR_PATIENT_AGE);
472                 sex = getAttrString(ATTR_PATIENT_SEX);
473         }
474
475         final protected void register()
476         {
477                 // registration is a way of telling scheduler what attributes are
478                 // needed for a message.
479
480                 // only the headers are registered
481                 // It is quite possible that the patient only has
482                 // had a subset of the tests. Registering
483                 // all tests would result in the dialog aborting
484                 // if any tests were missing.
485                 for(int i=0;i<TEST_HEADER_NAMES.length;i++)
486                 {
487                         registerAttribute(TEST_HEADER_NAMES[i]);
488                 }
489
490                 // to compute the normal for the AKALINE_PHOSPHATASE test
491                 // we need the age and sex of the patient.
492                 registerAttribute(ATTR_PATIENT_AGE);
493                 registerAttribute(ATTR_PATIENT_SEX);
494
495                 // register the prompts that we are using.  So that
```

```
496                     // they can be verified before the conversation begins.
497     //
498             //
499
500                     // register the roles that we require
501                     registerRole(ROLE_PATIENT);
502                     registerRole(ROLE_DOCTOR);
503             }
504
505             final protected void term()
506                     throws Exception
507             {
508             }
509
510             final protected void execute()
511             {
512                     // login has already been done.
513
514                     Phrase messageFrom = new Phrase();
515                     messageFrom.addMessage("Here are the results of your lab tests.");
516                     if(!getAcknowledgement(messageFrom))
517                     {
518                       setStatus(DIALOG_STATUS_FAIL);
519                       return;
520                     }
521
522                     Phrase abnormalHighTestExplanation = new Phrase();
523                     Phrase abnormalLowTestExplanation = new Phrase();
524                     boolean haveAbnormalTests = false;
525                     boolean haveAbnormalLowTests = false;
526                     boolean haveAbnormalHighTests = false;
```

```
527                 for(int i=0;i<testRows.length;i++)
528                 {
529 int compare = compareToNormal(i,testResult[i],testNormalLow[i],testNormalHigh[i]);
530
531
532                         if(compare == COMPARE_TEST_LOW)
533                         {
534                                 haveAbnormalLowTests = true;
535
536                         abnormalLowTestExplanation.
537 addMessage(ABNORMAL_LOW_TEST_EXPLANATION[i]);
538                         }
539                         else if(compare == COMPARE_TEST_HIGH)
540                         {
541                                 haveAbnormalHighTests = true;
542
543                                 abnormalHighTestExplanation.
544 addMessage(ABNORMAL_HIGH_TEST_EXPLANATION[i]);
545                         }
546                 }
547
548                 if(haveAbnormalHighTests || haveAbnormalLowTests)
549                         haveAbnormalTests = true;
550
551                 DialogTable dt = new DialogTable();
552                 if(isVerbose())
553                 {
554                         Phrase[] header = new Phrase[TEST_OUT_MAX];
555                         header[TEST_OUT_NAME]  = new Phrase(headerNames[TEST_FIELD_NAME]);
556                         header[TEST_OUT_RESULT]  = new Phrase(headerNames[TEST_FIELD_RESULT]);
557                         header[TEST_OUT_NORMAL_LOW]  = new Phrase(headerNames[TEST_FIELD_NORMAL_LOW]);
```

```
558                         header[TEST_OUT_NORMAL_HIGH]  = new Phrase(headerNames[TEST_FIELD_NORMAL_HIGH]);
559                         header[TEST_OUT_YOUR_RESULT]  = new Phrase(TEST_COLUMN_YOUR_RESULT);
560                         header[TEST_OUT_UNITS]  = new Phrase(headerNames[TEST_FIELD_UNITS]);
561                         header[TEST_OUT_REFERENCE]  = new Phrase(TEST_COLUMN_REFERENCE);
562
563                         dt.addHeader(header);
564                         for(int i=0;i<testRows.length;i++)
565                         {
566   Phrase[] row = new Phrase[TEST_OUT_MAX+1];
567
568   row[TEST_OUT_NAME]              = new Phrase(testRows[i][TEST_FIELD_NAME]);
569
570
571   row[TEST_OUT_RESULT]            = new Phrase(testResult[i]);
572
573   row[TEST_OUT_NORMAL_LOW] = new Phrase(testNormalLow[i]);
574
575   row[TEST_OUT_NORMAL_HIGH] = new Phrase(testNormalHigh[i]);
576
577   boolean isNormal = isTestNormal(i,testResult[i],testNormalLow[i],testNormalHigh[i]);
578
579   row[TEST_OUT_YOUR_RESULT] = new Phrase(isNormal);
580
581   row[TEST_OUT_UNITS]             = new Phrase(testRows[i][TEST_FIELD_UNITS]);
582   Phrase ref = choseReferencePhrase(i,testResult[i],testNormalLow[i],testNormalHigh[i]);
583
584   row[TEST_OUT_REFERENCE]     = ref;
585
586                                   dt.addRow(row);
587                         }
588                         getAcknowledgement(dt);
```

```
589                 }
590                 else
591                 {
592                         if(haveAbnormalTests)
593                                 getAcknowledgement(new Phrase("All your tests were normal with the exception of the following
594 tests."));
595                         else
596                                 getAcknowledgement(new Phrase("All your tests were normal. There were no abnormal results."));
597
598                         if(haveAbnormalTests)
599                         {
600                                 // terse mode - no headers
601                                 for(int i=0;i<testRows.length;i++)
602                                 {
603 boolean isNormal = isTestNormal(i,testResult[i],testNormalLow[i],testNormalHigh[i]);
604
605                                         if(! isNormal)
606                                         {
607         Phrase[] row = new Phrase[TEST_TERSE_OUT_MAX+1];
608
609         row[TEST_OUT_NAME]   = new Phrase(testRows[i][TEST_FIELD_NAME]);
610
611         row[TEST_OUT_RESULT] = new Phrase(testResult[i]);
612
613         row[TEST_OUT_UNITS]   = new Phrase(testRows[i][TEST_FIELD_UNITS]);
614
615
616                                                 dt.addRow(row);
617                                         }
618                                 }
619                         }
```

```
620                     getAcknowledgement(dt);
621             }
622             getAcknowledgement(new Phrase(TEST_EXPLANATION));
623             if(haveAbnormalLowTests)
624                     getAcknowledgement(abnormalLowTestExplanation);
625             if(haveAbnormalHighTests)
626                     getAcknowledgement(abnormalHighTestExplanation);
627     }
628
629
630     private Phrase choseReferencePhrase(int testIndex,double testResult, double normalLow, double normalHigh)
631     {
632             String ref = "";
633
634             int compare = compareToNormal(testIndex,testResult,normalLow,normalHigh);
635
636             if(compare == COMPARE_TEST_NORMAL)
637                     ref =
638 testRows[testIndex][TEST_FIELD_REFERENCE_NORMAL];
639             else if(compare == COMPARE_TEST_LOW)
640                     ref = testRows[testIndex][TEST_FIELD_REFERENCE_LOW];
641             else if(compare == COMPARE_TEST_HIGH)
642                     ref = testRows[testIndex][TEST_FIELD_REFERENCE_HIGH];
643             else if(compare == COMPARE_TEST_INDETERMINATE)
644                     ref = testRows[testIndex][TEST_FIELD_REFERENCE_INDETERMINATE];
645
646             return new Phrase(ref);
647      }
648
649      private boolean isTestNormal(int testIndex,double testResult, double normalLow, double normalHigh)
650      {
```

```
651     int compare = compareToNormal(testIndex,testResult,normalLow,normalHigh);
652             if(compare == COMPARE_TEST_NORMAL)
653                     return true;
654             return false;
655     }
656
657
658     private int compareToNormal(int testIndex,double testResult, double testNormalLow, double testNormalHigh)
659     {
660             int compare = 0;
661             if(testIndex == TEST_INDEX_ALKALINE_PHOSPHATASE)
662             {
663                     compare = compareForAlkalinePhosphatase(age,sex,testResult);
664             }
665             else
666             {
667                     if(testResult < testNormalLow)
668                             compare = COMPARE_TEST_LOW;
669                     if(testResult > testNormalHigh)
670                             compare = COMPARE_TEST_HIGH;
671             }
672
673             //
674             return compare;
675     }
676
677
678     private static int REFTABLE_COL_AGE_LOW         = 0;
679     private static int REFTABLE_COL_AGE_HIGH        = 1;
680     private static int REFTABLE_COL_VALUE_LOW               = 2;
681     private static int REFTABLE_COL_VALUE_HIGH      = 3;
```

```
682     private int[][] alkalinePhosphataseRefValByAgeMale =
683     {
684             // low age, high age, lowNormal, highNormal
685             { 1, 9, 145, 420 },
686             { 10, 11, 130, 560 },
687             { 12, 13, 200, 495 },
688             { 14, 15, 130, 525 },
689             { 16, 19, 65, 260 }
690     };
691
692     private int[][] alkalinePhosphataseRefValByAgeFemale =
693     {
694             // low age, high age, lowNormal, highNormal
695             { 1, 9, 145, 420 },
696             { 10, 11, 130, 560 },
697             { 12, 13, 105, 420},
698             { 14, 15, 70, 230},
699             { 16, 19, 50, 130}
700     };
701
702     // return 0 if normal, +1 for high, -1 for low and -2 for indeterminate
703     private int compareForAlkalinePhosphatase(int age,String sex,double testValue)
704     {
705             int[][] refTable = null;
706             if(sex.equalsIgnoreCase("male"))
707             {
708                     refTable = alkalinePhosphataseRefValByAgeMale;
709             }
710             else
711             {
712                     refTable = alkalinePhosphataseRefValByAgeFemale;
```

```
713                 }
714
715                 for(int i=0;i<refTable.length;i++)
716                 {
717                         if(age >= refTable[i][REFTABLE_COL_AGE_LOW] && age <=
718 refTable[i][REFTABLE_COL_AGE_HIGH])
719                         {
720                                 if(testValue <
721 refTable[i][REFTABLE_COL_VALUE_LOW])
722                                         return COMPARE_TEST_LOW;
723                                 if(testValue >
724 refTable[i][REFTABLE_COL_VALUE_HIGH])
725                                         return COMPARE_TEST_HIGH;
726                                 return COMPARE_TEST_NORMAL;             // must be normal
727                         }
728                 }
729
730                 return COMPARE_TEST_INDETERMINATE;              // not in the age range - indeterminate
731         }
732 }
733
```

EXHIBIT 3

EXHIBIT 3

PSEUDO CODE LISTING OF TEST RESULT DELIVERY FOR STREPOCOCCUS TEST

```
1  package com.solvetech.st_notify;
2
3  import java.util.*;
4
5  import com.solvetech.st_lang.*;
6
7  public class StrepTest_Dialog extends DialogBase
8  {
9
10         private final static String ATTR_DIALOG_YOUR_STREP_RESULTS = "dialog.strep_results";
11         private final static String ATTR_DIALOG_STREP_REFERENCE      = "dialog.strep_reference";
12         private final static String TEST_RESULT_MESSAGE_INTRO = "Your recent throat culture for Streptoccocus was";
13
14         private static final String ABNORMAL_TEST_EXPLANATION =
15                 "This germ can cause sore throat and fever and enlargement of neck glands. Even if you do not have these symptoms
16 you should receive an antibiotic to eradicate the germ so you do not have other complications.";
17
18         private static final String NORMAL_TEST_EXPLANATION =
19                 "This test showed no evidence of Streptococcus.  Other germs besides Strep can cause sore throat, fever and signs of
20 illness."
21
22         private static final String INDETERMINATE_TEST_EXPLANATION =
23                 "The test for Strep could not be processed because of laboratory technical difficulties or may take somewhat longer to
24 process.  If we are able to process it in spite of these difficulties, we will forward you the result in 24 hours.";
25
26         private boolean testResultPositive = false;
```

```
27          private String testResultReference = "";
28
29          //=================================================
30          public StrepTest_Dialog()
31          {
32
33                  super();
34          }
35          final protected void init()
36                  throws Exception
37          {
38          testResultPositive =
39  getAttrBoolean(ATTR_DIALOG_YOUR_STREP_RESULTS);
40                  testResultReference = getAttrString(ATTR_DIALOG_STREP_REFERENCE);
41          }
42
43          final protected void register()
44          {
45
46                  // registration is a way of telling scheduler what attributes are
47                  // needed for a message.
48
49                  registerAttribute(ATTR_DIALOG_YOUR_STREP_RESULTS);
50                  registerAttribute(ATTR_DIALOG_STREP_REFERENCE);
51
52                  // register the prompts that we are using.  So that
53                  // they can be verified before the conversation begins.
54
55
56                  // register the roles that we require
57                  registerRole(ROLE_PATIENT);
```

```
58                  registerRole(ROLE_DOCTOR);
59          }
60
61          final protected void term()
62                  throws Exception
63          {
64          }
65
66          final protected void execute()
67          {
68                  // login has already been done.
69
70                  Phrase testResultPhrase = new
71  Phrase(TEST_RESULT_MESSAGE_INTRO);
72
73                  if(testResultPositive)
74                          testResultPhrase.addMessage("POSITIVE");
75
76                          getAcknowledgement(new Phrase(ABNORMAL_TEST_EXPLANATION);
77
78                  if(testResultPositive = "NEGATIVE")
79                          testResultPhrase.addMessage("NEGATIVE");
80                          getAcknowledgement(new Phrase(NORMAL_TEST_EXPLANATION);
81
82                  if(testResultPositive = "INDETERMINATE")
83                          testResultPhrase.addMessage("INDETERMINATE");
84                          getAcknowledgement(new Phrase(INDETERMINATE_TEST_EXPLANATION);
85
86                                  if(!getAcknowledgement(testResultPhrase))
87                  {
88                    setStatus(DIALOG_STATUS_FAIL);
```

```
89                  return;
90                }
91
92                // for verbose mode, give the reference to Steptoccocus
93                if(isVerbose())
94                        getAcknowledgement(new Phrase(testResultReference));
95        }
96  }
97
```